FDU at TREC-10: Filtering, QA, Web and Video Tasks

Lide Wu, Xuanjing Huang, Junyu Niu, Yikun Guo, Yingju Xia, Zhe Feng
Fudan University, Shanghai, China

This year Fudan University takes part in the TREC conference for the second time. We have participated in four tracks of Filtering, Q&A, Web and Video.

For filtering, we participate in the sub-task of adaptive and batch filtering. Vector representation and computation are heavily applied in filtering procedure. Four runs have been submitted, which includes one T10SU and one T10F run for adpative filtering, as well as another one T10SU and one T10F run for batch filtering.

We have tried many natural language processing techniques in our QA system, including statistical sentence breaking, POS tagging, parsing, name entity tagging, chunking and semantic verification. Various sources of world knowledge are also incorporated, such as WordNet and geographic information.

For web retrieval, relevant document set is first created by an extended Boolean retrieval engine, and then reordered according to link information. Four runs with different combination of topic coverage and link information are submitted.

On video track, We take part in both of the sub-tasks. In the task of shot boundary detection, we have submitted two runs with different parameters. In the task of video retrieval, we have submitted the results of 17 topics among all the topics.

1. Filtering

Our research on filtering focuses on how to create the initial filtering profile and set the initial threshold, and then modify them adaptively. In this section, detailed introduction to the training and adaptation module of our adaptive runs is first presented. Then we introduced our batch runs briefly. Final part presents the experiment results.

1.1 Adaptive filtering

Figure 1.1 shows the architecture of the training in adaptive filtering. At first, feature vectors are extracted from positive and pseudo-positive document samples. The initial profile is the weighted sum of positive and pseudo-positive feature vectors. Then we compute the similarity between the initial profile and all the training documents to find the optimal initial threshold for every topic.

1.1.1 Feature selection

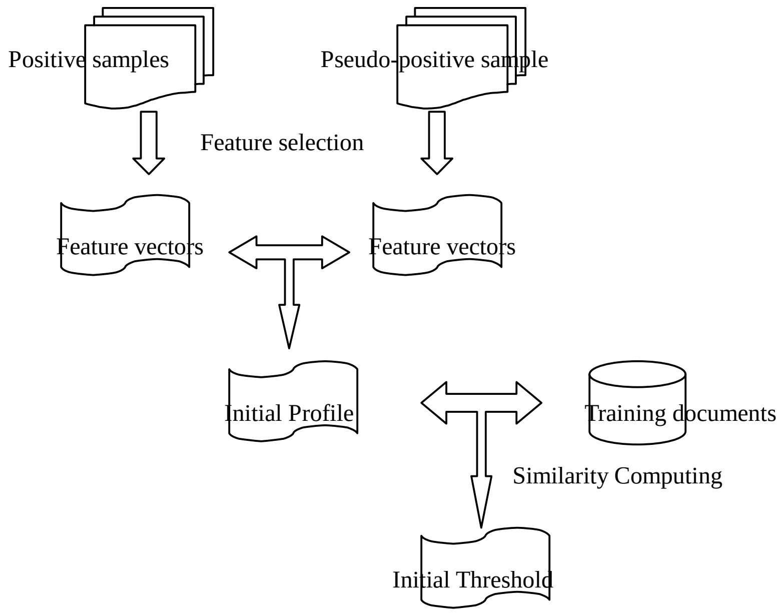
Since the total number of all words is very large and it costs much time in similarity computation, we decide to select some important words from them. First, we carry out morphological analysis and stopword removing. Then we compute the *logarithm Mutual Information* between remaining words and topics:

$$\log MI(w_i, T_j) = \log\left(P(w_i \mid T_j) \big/ P(w_i) \right) \tag{1.1}$$

Where, w_i is the ith word and T_j is the jth topic. Higher logarithm Mutual Information means w_i and T_j are more relevant. P(w_i) and P(w_i|T_j) are both estimated by *maximal likelihood method*.

For each topic, we select those words with logarithm Mutual Information higher than 3.0 and occurs more than once in the relevant documents. Logarithm Mutual Information is not only used as the selection criterion, but also as the weight of feature words.

Figure 1.1 Architecture of the training in adaptive filtering



1.1.2 Similarity Computation

The similarity between the profile and training documents is computed by the cosine formula:

$$Sim(d_i, p_j) = Cos\theta = \sum_k d_{ik} * p_{jk} \Big/ \sqrt{(\sum_k d_{ik}^2)(\sum_k p_{jk}^2)} \,. \qquad (1.2)$$

Where, p_j is the profile of the jth topic and d_i is the vector representation of the ith document. d_{ik}, the weight of the kth word in d_i, is computed as such: $d_{ik} = 1 + \log(tf_{ik} * avdl/dl)$, where tf_{ik} is the frequency of the kth word in the ith document , *dl* is the average number of different tokens in one document, *avdl* is the average number of tokens in one document.

1.1.3 Creating initial profile and Setting initial threshold

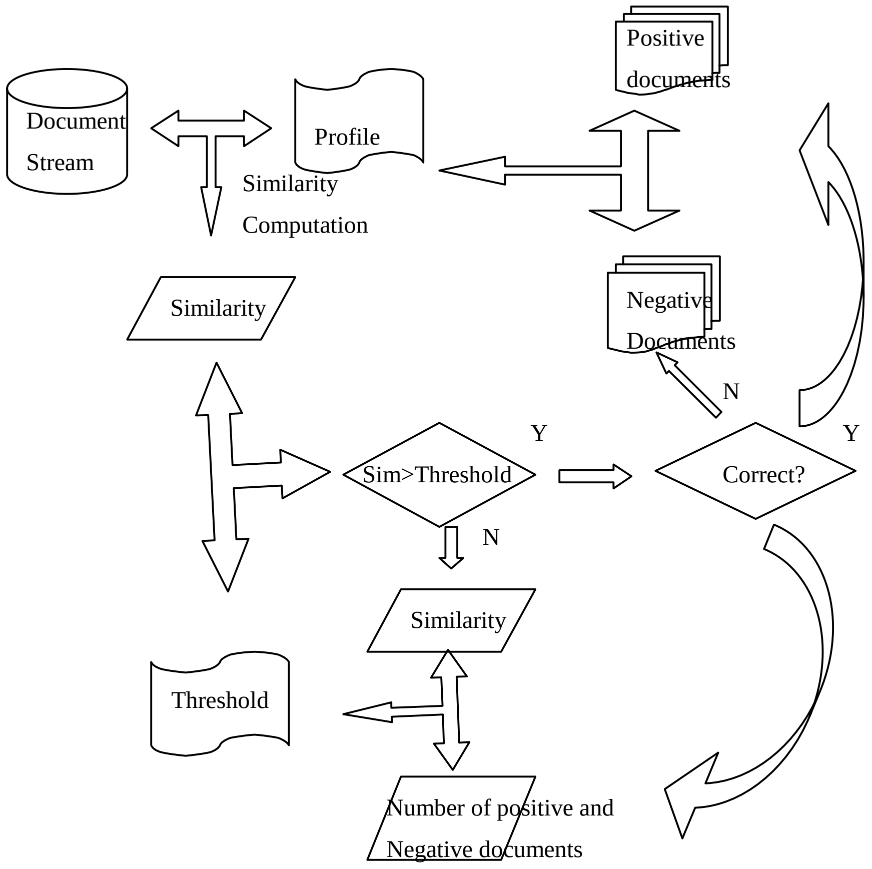
Each topic profile is represented by a vector which is the weighted sum of feature vector from positive (relevant) documents and feature vector from pseudo relevant documents with the ratio of 1: X_0.

To make use of the hierarchy of categories, those documents of the same high-level category are considered as pseudo relevant documents. Since the number of low-level categories is different among different high-level categories, we set different X_0 for different categories. In our experiment, X_0 is set to 0.03 for the topics from R1 to R79, and 0.15 for R80~R84. After combining the positive and pseudo-positive feature vectors, we get the initial profile. Once the initial profiles are acquired, the initial thresholds should be set to those values that can result in the largest value of T10U or T10F.

1.1.4 Adaptation of threshold and topic profile during filtering

For adaptive filtering, we adopt an adaptation procedure to modify the initial profile and threshold while filtering documents. Figure 1.2 shows the architecture for the adaptation:

Figure 1.2 Architecture for the adaptation



(1) Adjustment of threshold

We adjust the threshold once a positive document is retrieved. Let:

- *t*: denote the sequence number of document, since the documents are processed by temporal order, *t* also can be considered as time.
- $n(t)$: denote the number of documents processed up to *t*
- $n_R(t)$: denote the relevant documents retrieved up to *t*
- $n_N(t)$: denote the irrelevant documents retrieved up to *t*
- $T(t)$: denote the threshold at time *t*
- $S^-(t_k, t_{k+1})$: denote the average similarity of the document been rejected in (t_k, t_{k+1}) interval
- $P(t_k, t_{k+1})$: denote the precision of system in (t_k, t_{k+1}) interval, here

$$P(t_k, t_{k+1}) = {n_R(t_{k+1}) - n_R(t_k)} \Big/ {n(t_{k+1}) - n(t_k)}$$

Intuitionally, we should increase the threshold if the precision is too low and lower the threshold if too few documents are retrieved. So we can use $S^-(t_{k+1}, t_k)$ and $P(t_{k+1}, t_k)$ to decide whether to increase or decrease the threshold. When the precision is lower than expected, we should increase the threshold. Otherwise, we can decrease the threshold. In particular, when the threshold is too higher than the similarity with the rejected documents, the threshold should be decreased quickly. The above strategy of threshold adjusting can be written as below:

If $p(t_k, t_{k+1}) \le EP(t_{k+1})$ then

$$T(t_{k+1}) = T(t_k) + \alpha(t_{k+1}) \bullet (1 - T(t_k))$$

Else If $S^{-}(t_k, t_{k+1}) < T(t_{k+1}) * D$ then

$$T(t_{k+1}) = T(t_k) \bullet A + S^{-}(t_k, t_{k+1}) \bullet (1 - A)$$

Else $T(t_{k+1}) = (1 - \beta(t_{k+1})) * T(t_k)$

Where $\alpha(t_k)$ is the coefficient for increasing the threshold, and $\beta(t_k)$ is the coefficient for decreasing the threshold, both of them can be considered as the function of $n_R(t)$. In our experiment, we use the following linear functions shown in equation 1.3.

$$\alpha(t_k) = \begin{cases} \alpha_0 * (\mu - n_R(t_k))/\mu, & n_R(t_k) \le \mu \\ 0, & n_R(t_k) > \mu \end{cases}, \beta(t_k) = \begin{cases} \beta_0 * (\mu - n_R(t_k))/\mu, & n_R(t_k) \le \mu \\ 0, & n_R(t_k) > \mu \end{cases} \quad (1.3)$$

Where α_0 and β_0 are the initial parameter. The parameter of μ indicates the maximum number of positive documents should be used to adjust the threshold and modify the profile. Here we set $\alpha_0 = 0.02$, $\beta_0 = 0.1$ and $\mu = 300$.

The introduction of parameter D aims at increasing the recall. Since the actual number of relevant documents of every topic cannot be observed, we can only acquire some indirect estimation. We believed when the average similarity between the profile and those rejected documents are too small, the similarity threshold should be decreased in order to enhance the recall. In our experiment, we set D = 0.1 and A = 0.8.

$EP(t_k)$ means the precision which we wish the system to reach. At first, we regarded this parameter as constant and tried several different values, but the results are not very satisfactory. Since it is impractical to require the system to reach the desired high precision at the beginning of filtering, we adopt a gradual-ascent function. The function is showed in equation 1.4.

$$EP(t_{k+1}) = \begin{cases} P_0 + (P_{final} - P_0) * n_R(t_{k+1})/\mu, & n_R(t_k) \le \mu \\ 0, & n_R(t_k) > \mu \end{cases} \quad (1.4)$$

Where, P_0 and P_{final} are the desired initial and final precision. In our experiment, P_0 = 0.2 and P_{final} = 0.6.

(2) Adaptation of profile

Once a retrieved document has been judged relevant, it is added to the positive document set otherwise it is added to the negative document set. During profile adaptation, feature vectors are extracted from positive documents and negative documents. The new topic profile is the weighted sum of feature vector from positive documents and negative documents with the ratio of 1:X_1 (Here X_1= -0.25). For effectiveness and efficiency reason, we adjust the topic profile only after L(L = 5) positive documents have been retrieved.

1.2 Batch filtering

Since this year's batch filtering task does not include batch-adaptive task, there should be no adaptation in the batch-filtering sub-task. Therefore, the profile and threshold acquired from training should remain the same during filtering.

There is only a slight difference in the initialization module of our batch and adaptive runs. Full relevance judgments are provided in batch filtering. As a result, for batch run, the given relevant judgments are enough for us to build the initial profile, so pseudo-relevant documents are not used in profile creation. In addition, we adopt the stratified tenfold cross-validation method to avoid the phenomenon of overfitting.

1.3 Evaluation results

This year Fudan University has submitted four runs for adaptive and batch filtering. We submit no routing runs. Table 1.1 summarizes our adaptive and batch filtering runs. Four evaluation criteria are calculated, including *T10SU*, *T10F*, *Set Precision* and *Set Recall*. Underlined value means that the run is optimized for the corresponding criterion. The last columns give the number of topics in which our runs perform better, equal and worse than median ones according to the criteria for which our runs are optimized.

Task	*Run*	*T10SU*	*T10F*	*Set Precision*	*Set Recall*	*Comparison with median*		
						>	=	<
Adaptive	FDUT10AF1	<u>0.215</u>	0.404	0.505	0.330	64	5	15
	FDUT10AF4	0.213	<u>0.414</u>	0.493	0.363	71	4	10
Batch	FDUT10BF1	<u>0.248</u>	0.441	0.563	0.313	32	13	39
	FDUT10BF2	0.244	<u>0.448</u>	0.526	0.373	27	17	40

Table 1.1 Adaptive and batch filtering results

From this table we can find that our adaptive runs perform better than median for most of the topics, while our batch runs do not perform as well. Although our batch runs performs better than adaptive runs, the divergence is not very significant. It helps to show that adaptation plays a very important role in filtering.

2. Question Answering

It is the second time that we take part in the QA track. We tried many natural language processing techniques, and incorporated many sources of world-knowledge. A novel question answering technique, known as "*syntactic constrained semantic verification*", has been put forward. In next section, we will describe the architecture of our QA system, followed by a detailed discussion of the main components.

2.1 The Overview of QA system

Our system contains four major modules, namely question processing module, offline indexing module, online searching and concept filtering module, as well as answer processing module. The online models are represented in Figure 2.1.

Our indexing module creates full-text index for the document collection. However, it is quite different from traditional indexing procedure in that it incorporates several NLP techniques not only to avoid errors due to traditional stemming process, but also to increase both the precision and recall while retrieving proper name.

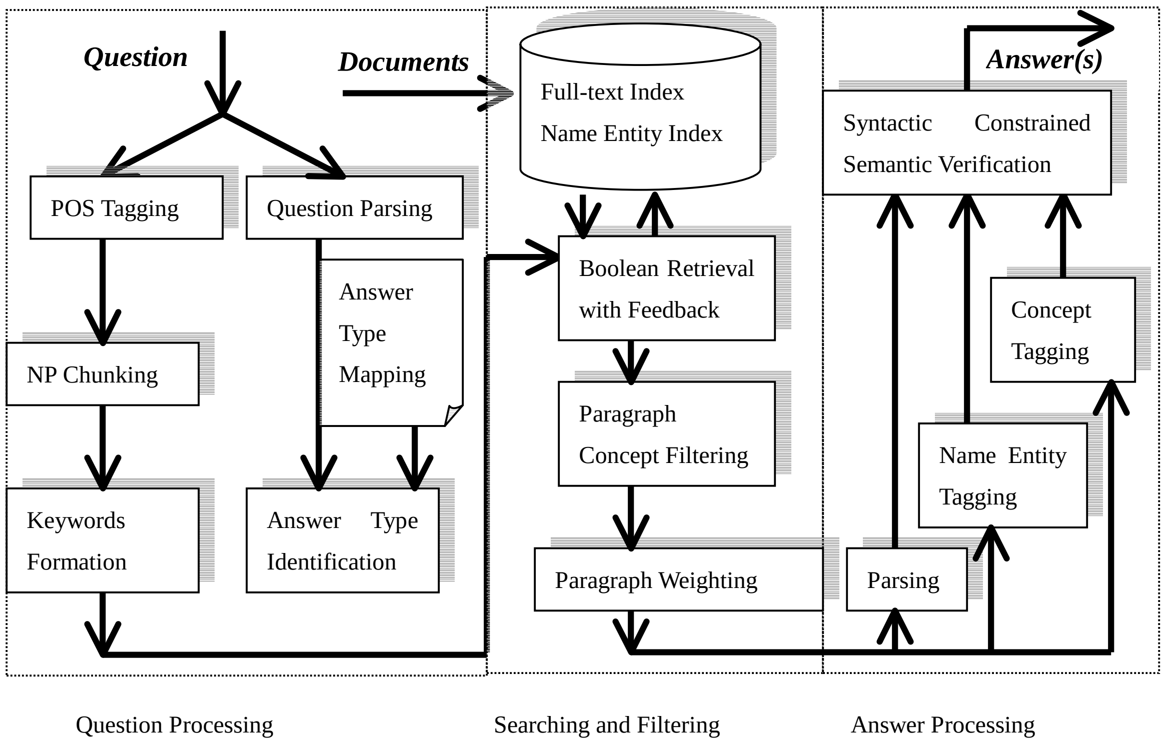
Question processing module tries to interpret the meaning of the input question by identifying answer type (the kind of information the question requires) from the question type, and extracting keywords. Next, the searching and filtering module use only non-replaceable keywords to retrieve relevant paragraph. After obtaining the result paragraphs, we use a concept thesaurus to filter and rank those paragraphs according to the number of occurring concepts, which are mainly derived from those replaceable keywords.

In the Answer Processing module, we use a dependency parser to analyze sentences in which the answer may lie in. Finally, a novel semantic verification scheme is applied after a WordNet-based concept tagging and a name entity tagging are completed.

2.2 Document Indexing

Our document indexing module actually includes two separate indices, i.e. a morphological analysis based full-text index and a proper name index. However, before we build these indexes, a sentence breaking module is applied to get correct sentence boundaries. We use a free sentence breaking tool, based on maximum entropy model, from Adwait Ratnaparkhi's web site.

Figure 2.1 Overview of our QA system (online part)



Proper name indexing is carried out to accelerate the online question processing speed. Our proper name tagging module depends heavily on a maximum entropy model based NP chunker. After reading each sentence with its part of speech tag (POS) for every word, it outputs NP chunks for the sentence. Figure 2.2 presents one sample sentence after NP chunking.

Figure 2.2 The output of NP chunking toolkit

[Ed Wilson] , [spokesman] for [the District] of [Columbia police] , said [street crime] in [Washington] has increased in [recent years] , but [there] have been [few reports] of [assaults] near [the Capitol grounds.]

2.3 Question Processing

The goal for question processing module is to find the user's information needs by examining the question. The query and expected answer type are transformed from every original natural language question.

2.3.1 Query Formation

First, considering synonyms, we define two kinds of keywords, i.e. the replaceable keywords and non-replaceable keywords. The replaceable keywords are referred to those words that could be replaced by other synonym without altering the information request of the question. Only the non-replaceable keywords are transformed into query. The documents returned by search engine will consist of all the candidate segments. Further, those candidates irrelevant to the question will be filtered out by replaceable keywords and their synonyms.

POS tagging and NP chunking are carried out to segment each question into segments. After that, we apply a HMM based Name Entity Identification tool to extract the non-replaceable keywords. It can recognize six kinds entity name, including people's name, place name, organization name, time, date and miscellaneous number, from normal NP phrases.

The NP phrases identified by Name Entity Identification module are regarded as the non-replaceable keywords and then submitted to the search engine, while other components are treated as the replaceable keywords.

2.3.2 Answer Type Concept Identification

Another task for Question Processing module is to determine the desired answer type concepts. First, we roughly classify 10 question types according to the question interrogatives, as shown in Figure 2.3. Next, 32 answer type concepts are introduced into our system, illustrated in figure 2.4. Among them, six are identified by Name Entity Identification tool, i.e. DATE, LOCATION, MONEY_UNIT, ORGANIZATION, PERCENTAGE and PERSON, while other concepts correspond to some synset in the WordNet noun hierarchies.

Figure 2.3 Question types

What	Who/whom	Where
How	why	how much
When	which	Name

Figure 2.4 Answer type concepts

ACTIVITY	COLOR	LINEAR_AMOUNT	PLANT
AMOUNT	COUNTRY	LOCATION	PRODUCT
ANIMAL	DATE	MONEY	PROVINCE
ARTIFACT	DEF	MONEY_UNIT	STAR
BODYPART	DISEASE	ORGANIZATION	TEMPERATURE
CAREER	ELEMENT	OTHER	TIME
CD	FOOD	PERCENTAGE	WEIGHT
CITY	LANGUAGE	PERSON	WORD

2.4 Search and Filtering Module

We employ the Boolean retrieval engine to find candidate answer paragraphs. We modify the search query to avoid returning too many and too few paragraphs. If too many paragraphs are retrieved, more keywords, such as replaceable keywords, will be added to restrict the number of candidate paragraphs. Otherwise, some of the key phrase will be removed.

After the paragraphs are retrieved, additional lexicon knowledge (Moby electronic thesaurus) is used to filter out irrelevant ones and sort remaining paragraphs according to the number of the words which appear in the question.

Moby electronic thesaurus contains about 1,000 concepts and each concept includes several words with similar word meaning [Moby00]. First, the replaceable keywords for each question are matched against the thesaurus to find one or more relevant concepts. Next, if the correspondent concept is found, the candidate paragraphs will be examined to find out the number of the words under the same concept. These words will be called extended query keywords (EQKs). Their number, which reflects the semantic closeness between a question and every paragraph, will be used to sort the paragraphs.

2.5 Answer Processing Module

We put forward a new approach in the Answer Processing module, which is named as "syntax constrained semantics verification". The Answer Processing Module aims to determine and extract answers from the candidate paragraphs retrieved by the Search and Filtering module. Figure 2.5 gives the framework of this module.

Firstly, we determine the answer type of every noun word and noun phrase in candidate paragraphs by Name Entity Tagging, which has been described before. The words whose answer types correspond to the Question Type are marked *candidate answer*.

Then the candidate paragraphs are passed through a dependency parser, Minipar [Lin98], to get the parsing tree. In this dependency tree, every node corresponds to a syntax category and every word in the candidate paragraph resides in a node. The children of a node are those words that modify it. We try to find a path in the parsing tree connecting EQKs and candidate answer. If there exist such path, we extract the words on the path and the children of them in the parsing tree. Thus we get different word groups for each candidate answers. Here we assume that these word groups are more semantically closer to the corresponding candidate answer they extracted from than other words in the same sentence.

Now we have a word group for each candidate answer. We firstly extract the content words (noun, verb, adjective and adverb words) from question to form another word group. Both word groups are considered to be relative to the focus words in question and answers. Then we compute their semantic similarity using a approach named *extended vector space model*. The result of this step is a similarity score which varies from zero to one. This is the basic factor to determine the final answer.

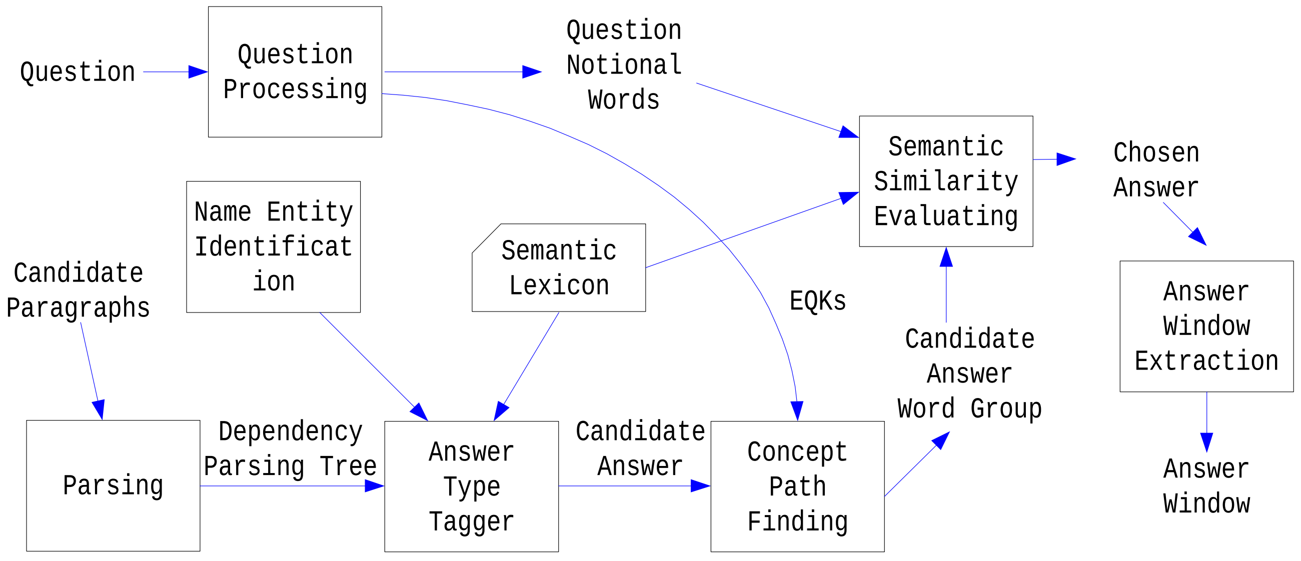


Figure 2.5 Syntax constrained semantic verification

For each candidate answer, a 50 byte-long section in the candidate paragraph, named *answer-windows*, is then created. This *answer-window* is centralized by the candidate answer. We evaluate each answer-window using the following three scores:

- *Semantic similarity:* This score has been computed using *extended vector space model*.
- *Syntax pattern score*: This score is based on the candidate answer sentence's syntactic structures. It takes several syntactic features into consideration, such as the length of the path in the parsing tree between the answer keywords and candidate answer, POS of EQKs and candidate answer.
- *Indicators score:* Some phrases or words, such as 'known as', 'called', 'named as' and some syntax features, such as appositive, strongly indicate a possible answer to specified questions such as 'who', 'what', 'which'.

The final score is a linear combination of all of these scores, where the weight of every score depends on the question feature.

2.6 Experiment results

This year we only take part in the main task of QA, and only submit one 50-byte run. Our results are not satisfactory. Statistics over 492 questions shows the strict mean reciprocal rank of 0.137 and lenient mean reciprocal rank of 0.145. Almost 80% of the questions return no correct response.

3. Web Retrieval

This year we attend the TREC-10 web subtask for the first time. We submit four runs for the web track. We used different combination of information in the four runs: title only and content only (fdut10wtc01), title with description and narrative information and content only (fdut10wac01), title only with link information (fdut10wtl01) and all title, description, narrative with link information (fdut10wal01).

3.1 System Architecture

In order to get better performance on web information retrieval, we have modified most of our original search engine, which is based on statistical model, and make a new search kernel that is based on extended Boolean search. Moreover, we split the document set and indices into several parts to efficiently handle the corpus of WT10g which contains 10G HTML documents.

Figure 3.1 shows the architecture of our web retrieval system. The first part in the left side is preprocessing module which can turn HTML pages into plain text. The second module is a preparation part for indexing, it combine all the small HTML documents in one directory of WT10g into a single file. The next step is indexing. We use stopword removing and morphology analysis to select entries of the indexing lexicon. What's more, we do not create index for the whole WT10g corpus due to the huge size of the corpus. Instead, the corpus is split into smaller parts, each of which is to be indexed independently. By this means, we can control the indices more easily than simply creating a larger index of the whole corpus.

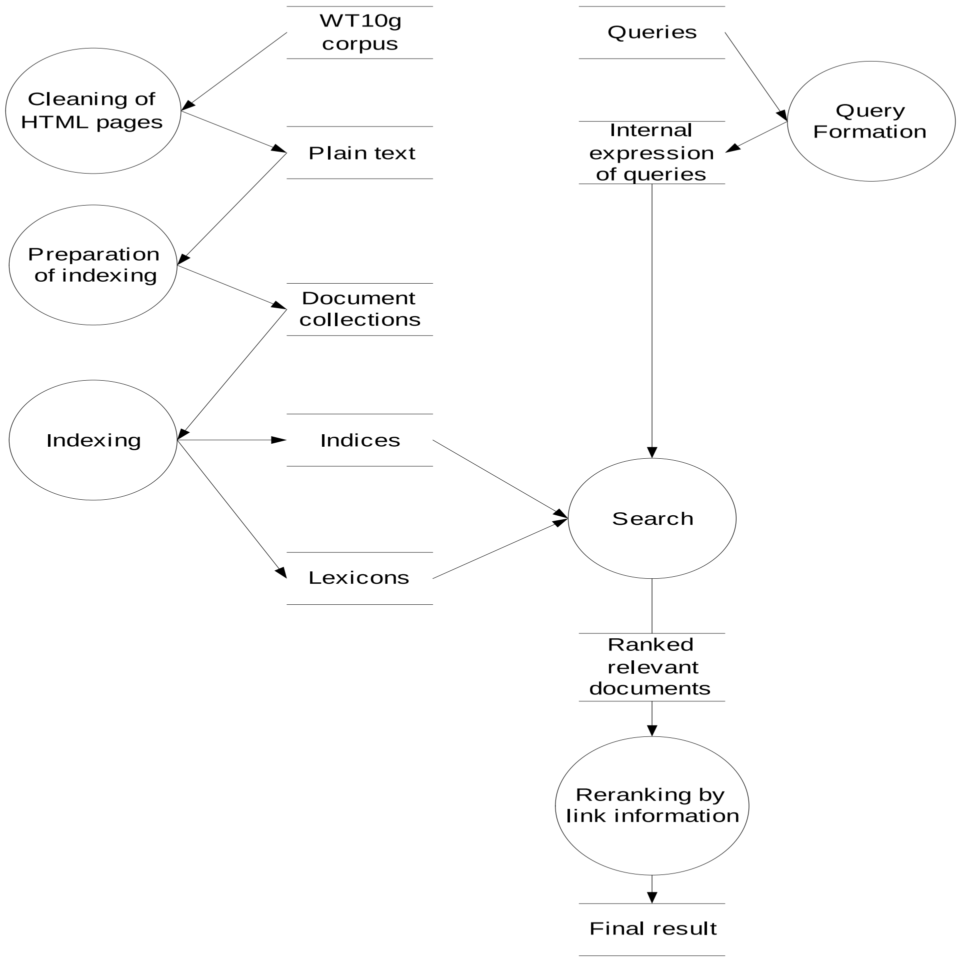
On the right side, the first step transfers the queries into several words, recognizes the phrase, and does some query expansion. The second module searches the index with the algorithm below and builds the ranked relevant document set. In the third step, link information is exploited to reorder the relevant documents.

3.2 Search Algorithm

The core algorithm is based on an extended Boolean retrieval engine called "short matching passages" [Kleinberg98], which intends to find the shortest passage that matching certain words. The assumption is that the shorter the passage is, the more possible that it will be relevant with the query. The following equation shows the basic idea of this algorithm. The *I*(*p*,*q*) represents the intensity of a shortest passage from the pth to qth word which contains certain keywords.

$$I(p,q)=\begin{cases}\left(\dfrac{K}{q-p+1}\right)^{a}, & \text{if q - p} + 1 \geq \text{K} \\ 1, & \text{otherwise}\end{cases} \tag{3.1}$$

Figure 3.1 Architecture of our web retrieval system



Before searching, natural language topics are first changed into Boolean queries, then the retrieval engine searches on all the individual indices simultaneously. Instead of the original equation in [Gordon98], we use our own equation (3.2) to calculate the score of each document.

$$S(D)=\sum_{(pi,qi)\in D} I(p_i,q_i)*w1(p_i,q_i) \tag{3.2}$$

Together with the length of short passage, we also consider the position of them (which is calculated in w1 in the equation above) to calculate the score of the document.

3.3 Reordering with link information

In order to improve Retrieving result by link information, we have tested Kleinberg Algorithm of hubs and authorities [Kleinberg98], co-citation and bibliographic coupling [Kraaij99]. After some experiments, we find that co-citation and bibliographic coupling can lead to the batter result. However, our best result still shows that link information cannot improve the search result.

The basic principle of co-citation is that if two documents (document A and B) cite the same documents, then document A and B are similar to some degree. The basic principle of bibliographic coupling is that if many documents cite both document A and B, then document A and document B are similar to some degree.

In our experiment, we use the formula of Wessel Kraaij [Kraaij99]:

$$Inlinkrel(d) = \frac{\sum_{i \in inlinks(d)} relevancy(i)}{\# inlinks(d)}, \quad Outlinkrel(d) = \frac{\sum_{i \in outlinks(d)} relevancy(i)}{\# outlinks(d)}$$

$$Citerel(d) = \frac{\sum_{i \in inlinks(d)} Outlinkrel(i)}{\# inlinks(d)}, \quad Bibcouplrel(d) = \frac{\sum_{i \in outlinks(d)} Inlinkrel(i)}{\# outlinks(d)}$$

Now, to the dth document, we have five scores: *S(D), InlinkRel(d), OutlinkRel(d), CociteRel(d), BibcoupleRel(d).* We can use the following formula to calculate the news core:

$$NewScore(d) = \alpha_1 * S(D) + \alpha_2 * InlinkRel(d) + \alpha_3 * OutlinkRel(d) + \alpha_4 * CociteRel(d) + \alpha_5 * Bibcouple(d) \qquad (3.3)$$

Where, $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ are five parameters.

3.4 Experiment Results

We submitted four runs, whose names are *fdut10wtc01, fdut10wtl01, fdut10wac01, fdut10wal01.* The final result is shown in the following table.

Runs	Type	Average Precision	R-Precision
Fdut10wtc01	*Title-only/content-only*	0.1661	0.2061
Fdut10wtl01	*Title-only/content + link*	0.1544	0.1939
Fdut10wac01	*Title + Description + Narritive/content-only*	0.1661	0.2061
Fdut10wal01	*Title + Description + Narritive/content + link*	0.1248	0.1607

Table 3.1 Results of web track

We used many different combinations of parameters to reordering the content-only result, and find finally that $\alpha_1=1$, $\alpha_2=0$, $\alpha_3=0$, $\alpha_4=1$, $\alpha_5=0$ can lead to the best result. But it still does not improve the content-only score.

4. Video Track

On Video Track, we take part in both Shot Boundary Detection and Video Retrieval. In the task of Shot Boundary Detection, we have submitted two runs with different parameters. One of them is precision-orientied, and another is recall-orientied. In the task of Video Retrieval, we submitted the results of 17 topics out of 74.

4.1. Shot Segmentation

In our system, we use FFD (*Frame-to-Frame Difference*) [Hanjalic97] to detect the shot boundary. But we redefined the difference between the *n*th frame and (*n*+*k*)th frame as Equation 4.1.

$$Z_k(n) = D_k(n) \times Y_k(n) \tag{4.1}$$

Where, $D_k(n)$ describes the difference on the luminance and $Y_k(n)$ describes the difference on the Color Histogram. The definitions of $D_k(n)$ and $Y_k(n)$ are as follows:

$$D_k(n) = C_1 \sum_i \sum_j | I(i, j, n+k) - I(i, j, n) | \tag{4.2}$$

$$Y_k(n) = 1 - C_2 \sum_i \min(H(i, n+k), H(i, n)) \tag{4.3}$$

Where, $I(i, j, n)$ is the average luminance of block (*i, j*) in frame *n*, and each block has 8*8 pixels. $H(i,n)$ is the Color Histogram value of frame *n* on the *i*th bin.

We use $Z_1(n)$ to detect the Cut Shot Boundary and $Z_k(n)$ to detect the Gradual Shot Boundary. θ_C and θ_G are thresholds for cut and gradual shot boundary. They will be discussed in next paragraph. If $Z_1(n)$ exceed the threshold θ_C, it may be caused by Cut Shot Boundary from frame *n* to frame *n*+1. We have also trying to reduce the influence of flashlight by compare the frames of both sides. When flashlight comes, it can be assumed that there will be more than one frame whose $Z_1(n)$ is larger than θ_C, also, the frames before and after the flashlight are similar. The Gradual shot boundary cannot be easily detected by the FFD of two continuous frames. We use $Z_k(n)$ (*k=50*) to magnify the frame-to-frame difference. But, it will cause more false alerts by motion in shot. To reduce that, we use motion detection: a sequence of continuous frames whose FFD is larger than θ_G will be labeled as Gradual Shot Boundary only if no efficient camera motion is detected and the frames before and after the sequence are dissimilar.

During the Shot Boundary Detection, threshold θ_C and θ_G are selected automatically every 500 frames [Zhu00]. The selection is according to the histogram of $Z_1(n)$ and $Z_k(n)$ in 500 frames. The histogram of $Z_1(n)$ and $Z_k(n)$ are calculated, and we find the first low point *p*. The value on *p* will be the threshold.

Other parameters are adjusted manually based on the 42 training video clips. The results show that the parameter selection is not sensitive in our method.

4.2. Video Retrieval

We submitted the results of 17 topics in video retrieval. Table 4.1 shows the type of these topics.

Topic Type	General	Known-Item	Total	Topic No.
People Searching	3	7	10	20,21,22,23,24,34,35,36,42,58
Object Serching	1	1	2	54,69
Video Text Searching	0	1	1	70
Camera Motion Searching	2	0	2	44,74
Shot Change Type Searching	1	0	1	65
Searching based on Document	0	1	1	62

Table 4.1 Submitted Topics

In Section 4.2.1, we will describe the architecture of our video retrieval system. Section 4.2.2 is the detailed description about implementation.

4.2.1 System Architecture

Our Video Retrieval System includes two parts. One is the off-line Indexing Sub-system, and another is on-line Searching Sub-system. Figure 4.1 describes the system architecture.

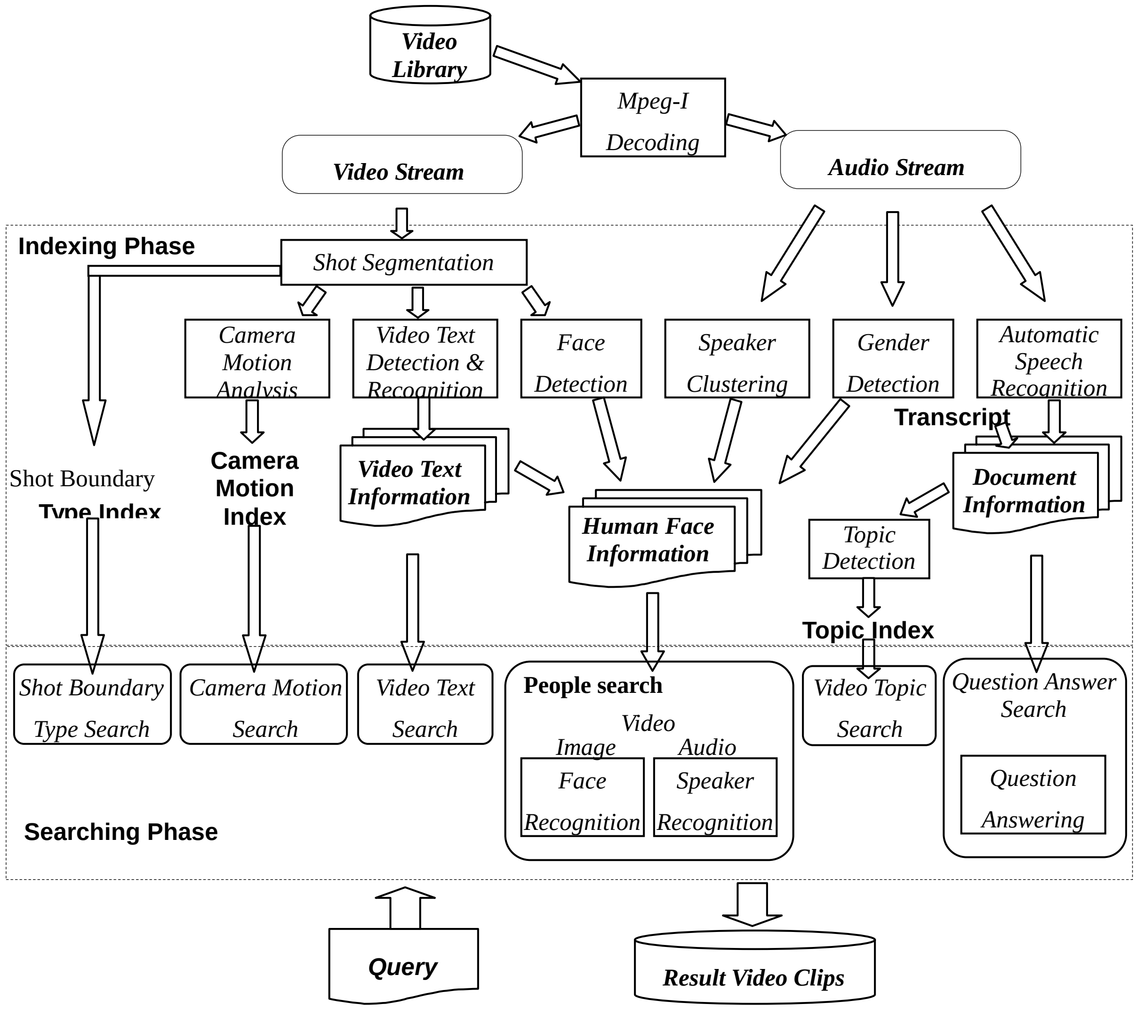


Figure 4.1 System architecture of video retrieval

4.2.2 Detailed Implementation

4.2.2.1 Qualitative Camera Motion Analysis

In our system, we analyze Camera Motion by the Motion Vectors obtained from MPEG stream. Each motion is composed of Motion Amplitude and Motion Direction. The system tries to segment shots into sub-shots automatically. We define sub-shot as some continuous frames in one shot with the similar camera motion.

4.2.2.2 Face Detection

Our method is designed mainly for interviewee detection. It has features: (1) The face is quite large, (2) the face has motion but the background is still. The method consists of three steps: Skin-Color based

Segmentation, Motion Segmentation, and Shape Filtering.

In skin-color based segmentation, we use several skin-color filters in different color spaces and combine them by AND operation. It is found that the result is better than the ones in any single color space. After that we have several skin-color regions. Similarly, we can have several motion regions by motion segmentation. By INTERSECTION operation, we have those regions which have both skin-color and motion. They are the candidates of face. For these candidates we use shape filtering to remove those too small or irregular ones and get the final results.

4.2.2.3 Face Recognition

In the training phase, we normalize all the training samples to 40*40 and make histogram equalization. After that, we clustering some of these samples and transform the face images to column vectors. Then:

(1) Let $\omega_1, \omega_2, \cdots, \omega_m$ be *m* training pattern classes, which correspond to *m* persons, respectively. Then calculate the within-class scatter matrix S_w of $\omega_1, \omega_2, \cdots, \omega_m$ and the covariance matrix S_t of all of samples.

(2) Calculate the zero subspace of the within-class matrix S_w .

(3) Calculate the eigenfaces in subspace $S_w^{-1}(0)$.

Suppose $S_w^{-1}(0) = span\{\alpha_1, \alpha_2, \cdots, \alpha_k\}$, where $\alpha_1, \alpha_2, \cdots, \alpha_k$ are the orthogonal unit vectors. Then, an arbitrary vector φ in $S_w^{-1}(0)$ can be represented as Equation 4.4:

$$\varphi = \xi_1\alpha_1 + \cdots + \xi_k\alpha_k = PZ \qquad (4.4)$$

where, $P = (\alpha_1, \alpha_2, \cdots, \alpha_k)$, $Z^T = (\xi_1, \xi_2, \cdots, \xi_k)$

Define matrix S_{t1} as $S_{t1} = P^T S_t P$. Therefore, the calculating of eigenface in subspace $S_w^{-1}(0)$ can be transformed to the problem calculating the eigenvectors corresponding to the *l* largest eigenvalues of the matrix S_{t1}. Suppose $\beta_1, \beta_2, \cdots, \beta_l$ are the *l* orthogonal unit eigenvectors corresponding to the *l* largest eigenvalues of the matrix S_{t1}. Then the required eigenfaces are $P\beta_1, P\beta_2, \cdots, P\beta_l$.

In recognition phase, we also normalize the Example Face and the Testing Face to 40*40 at first. Then we make histogram equalization and transform them to column vectors x_1, x_2 . Then the projection feature vectors of two images on the eigenface space can be obtained as Equation 4.5:

$$\xi_1 = (P\beta_1, P\beta_2, \cdots, P\beta_1)^T x_1, \quad \xi_2 = (P\beta_1, P\beta_2, \cdots, P\beta_1)^T x_2, \qquad (4.5)$$

Considering the distance $d = \|\xi_1 - \xi_2\|$, if $d \le \delta$, then we think that the Testing Face and the Example Face belong to same person. (δ is a threshold)

4.2.2.4 Video Text Detection and Recognition

There are three main parts in our Video Text Detection System: Text Block Detection, Text Enhancement and Binarization. To reduce the processing time, the system processes only one frame in every ten. On each processed frame, text block detection is first applied to get the position of each possible text line. This detection is based on edge image. Since edges are not sensitive to intensity changes and edges are dense in text line blocks, we calculate the gray scale edges in RGB space horizontally. The edge images are then binarized and run length analysis is applied to find candidate text blocks.

We use SSD (Sum of Square Difference) based block matching to track the detected text lines. All tracked text blocks are interpolated to a reasonable fixed size and then registered. At last, the tracked, interpolated and registered text blocks are combined by an average operation to reduce the noise and suppress the complex background.

An improved logical level technique (ILLT) is developed to binarize each candidate text block. This method can deal with different intensities of characters (i.e. characters may be brighter or darker than background) efficiently.

We have used commercial software: *TextBridge Pro Millennium* to recognize the binarized text block image. At last, the recognized string of each text blocks will be split into words and those words that are too short (less than 3) are removed in the filtering step.

4.2.2.5 Speaker Recognition and Speaker Clustering

During the Indexing Phase, Speaker Clustering can ensure the clustered shots that include human face are from the same person. And during the Searching Phase, Speaker Recognition can ensure the audio contained in the returned results is the same as the audio examples.

We use Vector Quantization (VQ) and Gaussian Mixture Model (GMM) to characterize each speaker. In these two methods, VQ worked faster than GMM while GMM performed better than VQ in some cases. In addition, in order to remedy the disturbance of noise and other backgrounds, a global model was used to modify the output of single speaker model.

Both VQ and GMM, especially after being integrated with global model, showed satisfactory detection and clustering effect if the speeches have the similar backgrounds. But if there is strong music background in the speech, the result will be worse.

4.2.2.6 Automatic Speech Recognition

We have used the Speech SDK of Microsoft as Automatic Speech Recognition (ASR) engine [www.Microsoft.com/speech]. There're different parameter sets for male and female speaker. One has to choose a proper parameter set to get better performance. And the engine has capability of background adaptation.

To increase the recall, we did not use gender detection. Instead, we use different parameter sets to recognize the same piece and give the confidence and time alignment of every recognized word. We have tried speaker change detection and audio classification. However, there is little improvement. The main reason is that the background music is too strong. Nevertheless, it recognized most of the keywords correctly and we use it for Topic Detection.

For the NIST video, we find that there are some errors in the human transcripts. We use ASR engine to give the result with time alignment. Then the result is aligned with the human transcript by ASR evaluation program. Finally, we get the rough time alignment between the audio and the human transcript. This time alignment is used for Question Answering.

4.2.2.7 Gender Detection

In our system, Gender Detection is made by Gaussian Mixture Model (GMM). We select pieces of audio that contains low background noise or music from the unused videos to train a Male Model and a Female Model. The feature is 12-dimension LPC cepstrum. Each model consists of 128 mixtures. Because the "clean" data of female is not enough, some male speakers are recognized as female. On the other hand, this error is also caused by the background music.

4.2.2.8 Topic Detection and Question Answering

In order to make Topic Detection and Question Answering, we should have a document library at first. In our system, the documents of videos are obtained in two ways. One is the manually created information. Another is the transcript created by automatic speech recognition (ASR). After that, Topic Detection and Question Answering module will run. These two modules come from Filtering and Question Answering, which can be found in Section 1 and 2. The training data of Topic Detection comes from the unused videos.

4.3. Results

Our results of shot boundary detection and video retrieval are presented in the following tables. As for shot boundary detection, our system acquires high performance on cut shot, while the results of gradual shot are not very good. Our performance of know-item search and general search both seems satisfactory. The reason may be attributed to the fact that we only submit the results of 10 know-item search topics and 7 general search topics.

Precision Oriented	Insert Rate	Delete Rate	Precision	Recall	Recall Oriented	Insert Rate	Delete Rate	Precision	Recall
Cut	0.039	0.028	0.961	0.972	Cut	0.039	0.028	0.961	0.972
Gradual	0.322	0.415	0.737	0.584	Gradual	0.350	0.391	0.727	0.608
All	0.133	0.154	0.889	0.845	All	0.143	0.146	0.883	0.853

Table 4.2 Shot Boundary Detection Results

Submitted Topics Number : 7	
Mean Precision	0.640

Table 4.3 General Search Results

Submitted Topics Number : 10				
	KI = 0.333 RI = 0.333	KI = 0.333 RI = 0.666	KI = 0.666 RI = 0.333	KI = 0.666 RI = 0.666
Mean Precision	0.543	0.539	0.434	0.430
Mean Recall	0.678	0.636	0.528	0.486

Table 4.4 Known-Item Search Results

ACKNOWLEDGMENTS

This research was partly supported by NSF of China under contracts of 69873011 and 69935010. We are thankful to Lin Mei, Yuefei Guo, Weixiang Zhao Yi Zheng, Yaqian Zhou, Kaijiang Chen, Xiaoye Lu, Jie Xi, He Ren, Li Lian, Wei Qian, Hua Wan and Tian Hu for their help in the implementation.

Reference

[Hanjalic97] Alan Hanjalic, Macro Ceccarelli, Reginald L.Lagendijk, Jan Biemand: *Automation of Systems Enabling Search on Stored Video Data*.Proc.SPIE, Vol.3022 of Storage and Retrieval for Image and Video Database V, 1997

[Harabagiu99a] Sanda Harabagiu and Dan Moldovan. *A Parallel System for Textual Inference*. IEEE Transactions parallel and distributed systems, vol.10, 1999

[Harabagiu99b] Sanda Harabagiu and Dan Moldovan. *FALCON: Boosting Knowledge for Answer Engines*. CSE, Southern Methodist University, 1999

[Kleinberg98] Jon M. Kleinberg : *Authoritative Sources in a Hyperlinked Environment*, Proc. 9th ACM-SIAM Symposium on Discrete Algorithms 1998

[Kraaij99] Wessel Kraaij, Thijs Westerveld: *TNO/UT at TREC-9: How different are web documents*, Proceeding of TREC-9.

[Lin98] Dekang Lin. *A Dependency-based Method for Evaluating Broad-Coverage Parsers*. Natural Language Engineering. 1998.

[Moby00] http://www.dcs.shef.ac.uk/research/ilash/Moby/

[Zhu00]Xingquan Zhu, Xiangyang Xue, Lide Wu: *An Automatic Threshold Detection Method in Video Shot Segmentation*, Vol.37 No.1, Chinese Journal of Computer Research and Development, 2000

[Gordon98] Gordon V.Cormack, Christopher R.Palmer, Michael Van Biesbrouck, Charles L.A. Clarke. *Deriving Very Short Queryies for High Precision and Recall (MultiText Experiments for TREC-7)*, Proceeding of TREC-7.